Wrocław University of Science and Technology

Laboratory of Optoelectronics and Photonics

Chair of Electronic and Photonic Metrology

# MULTI-SENSOR OPTOELECTRONIC CHRONOMETER WITH LASER TRIGGERING AND REAL-TIME DATA LOGGING

Optoelectronics Project - First Report

Group: E-61

Tuesday 7.30-9.00

Azra Selvitop #276772

Yiğit Temiz #276710

Wroclaw 2024

# Table of Contents

# 1 Project Description

The objective of this project is to develop a chronometer that accurately tracks the timing of competitors in races, such as swimming or running events. The chronometer will start when a racer passes through a laser, triggering a proximity sensor that detects the motion. It will continue to track the time until another object is detected by the sensor. Our project aims to provide more precise timing in races compared to human eyes.

# 2 Theoretical Introduction and methods

To achieve the objective of accurately tracking race times, we will use an optoelectronic setup, a microcontroller-based processing unit, and visual indicators to create a robust and precise chronometer. The following methods outline how these components will work together to detect race start and stop events, display timing information, and ensure system reliability.

## Chronometer Triggering and Detection

- Laser Beam Setup: The KY-008 Laser Transmitter Module will emit a focused laser beam across a predefined path (e.g., starting or finishing line), acting as the primary detection mechanism.
- Timing Start and Stop Mechanism: When a competitor crosses the starting line, they break the laser beam emitted by the KY-008. This interruption is detected by the microcontroller through signal processing, triggering the timer to start.
- Stop Time Detection: At the finish line, another KY-008 module is used. The interruption of this second laser beam signals the microcontroller to stop the timer, completing the measurement.

## Data Processing with ATMEGA328 Module

- Signal Processing: The ATMEGA328 microcontroller will interpret signals from the KY-008 laser module tracking both start and stop events. The CH340 AVR EDU ensures compatibility with Arduino IDE, simplifying programming and data handling.
- Time Calculation and Storage: The microcontroller records the time elapsed between start and stop signals, storing it in memory for display and later retrieval if required.

## Timing Display

- 1602 2x16 LCD Display with I2C IIC HD44780 Converter: The recorded time will be displayed in real-time on the LCD, providing clear visibility for judges or officials. The I2C connection allows for streamlined communication between the microcontroller and the display, minimizing pin usage and ensuring efficient data transfer.
- Displaying Relevant Information: The LCD can display the timer status, showing “Ready,” “Timing,” and the final “Stop” time, along with the recorded time in seconds for easy reading.

### Optoelectronic Feedback Mechanism

- LED Status Indicators: Colored LEDs are used to give visual cues on the chronometer's current state:
- Green LED: Signals that the chronometer is ready to start timing.
- Yellow LED: Lights up during timing to indicate an active measurement.
- Red LED: Lights up to signal the end of timing, confirming that the chronometer has recorded the stop time.

### Breadboard Assembly

- Breadboard Setup: The prototype board provides a structured environment for assembling the circuit, connecting components such as the photodiode, photodetector, LCD display, LEDs, and microcontroller.
- Male-Male Jumper Cables: These will be used to create connections between the components on the prototype board, allowing for easy modifications and troubleshooting during development.

### Calibration and Testing

- Adjusting the Photodiode and Photodetector Sensitivity: Before use, the photodiode and photodetector will be calibrated to ensure accurate detection of the competitor at start and finish lines.
- Test Runs: Several trial runs will be conducted to confirm the accuracy and consistency of the timing. Adjustments will be made to optimize performance under race conditions.

By implementing this setup, the chronometer will provide reliable, accurate, and automated race timing, offering an improvement over manual timing methods and enhancing precision in competitive events.

# 3 Equipment Used

- Breadboard
- 1602 2x16 LCD display with I2C IIC HD44780 converter
- ATMEGA328 Module Compatible with Arduino UNO CH340 AVR EDU + GOLDPIN Uno R3
- Male-Female Jumper cables
- The KY-008 Laser Transmitter Module
- Colored LED (Red, Yellow, Green)
- 200 Ohm resistor (x3)

# 4 Assumptions

In this section, the functional and design assumptions of the project are listed in tabular form format.

Table 4.1: Functional and design assumptions.

| Functional Assumptions | Design Assumptions |
|---|---|
| The system accurately tracks the start and stop times of competitors in races by detecting interruptions in a laser beam. | KY-008 emits a continuous laser beam that acts as the triggering mechanism. The system detects interruptions in this beam through microcontroller signal processing. |
| Providing real-time visual feedback on its status using colored LEDs (green for "ready," yellow for "timing," and red for "stopped"). | Arduino UNO processes signals from the KY-008 laser module to start and stop the timer and controls the LED indicators and LCD display. |
| The elapsed time is displayed clearly on LCD | LCD is integrated via an I2C interface to display the recorded time in real-time with minimal latency. |
| The chronometer operates reliably under varying environmental conditions typical for race events. | The components are assembled on a prototype board using jumper wires for easy testing, calibration, and troubleshooting. |

# 5 Hardware

This chapter covers the mechanical and electrical elements that were used for the creation of the project.

This image below represents the idea of the system of our project.

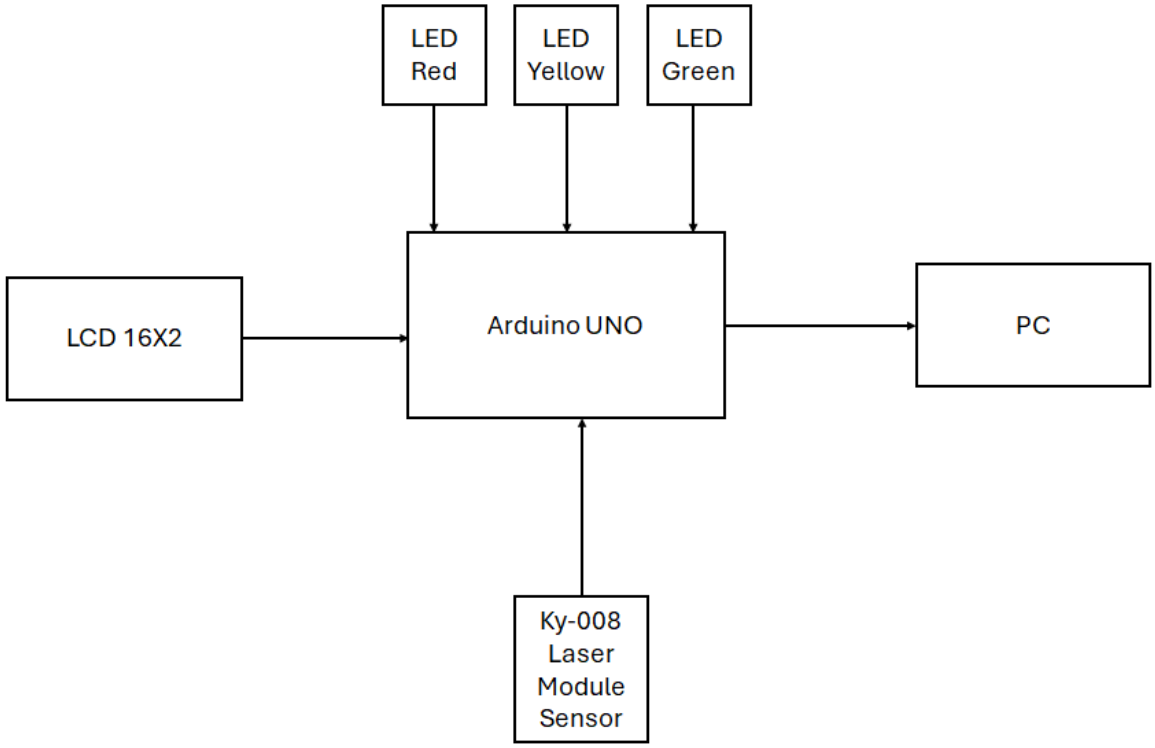


Figure 5.1: Block diagram of the system.

Explanation: The system consists of an Arduino UNO, which processes inputs from a KY-008 Laser Module Sensor and controls the outputs displayed on an LCD and three status LEDs (red, yellow, green). The laser sensor detects events and signals the Arduino, which then calculates and tracks the elapsed time.

The green, yellow, and red LEDs indicate system status during different stages of operation. The data, including the elapsed time, is displayed on the LCD and can be transmitted to a PC for further analysis if needed. The system provides real-time feedback and visual indication of the timing process.

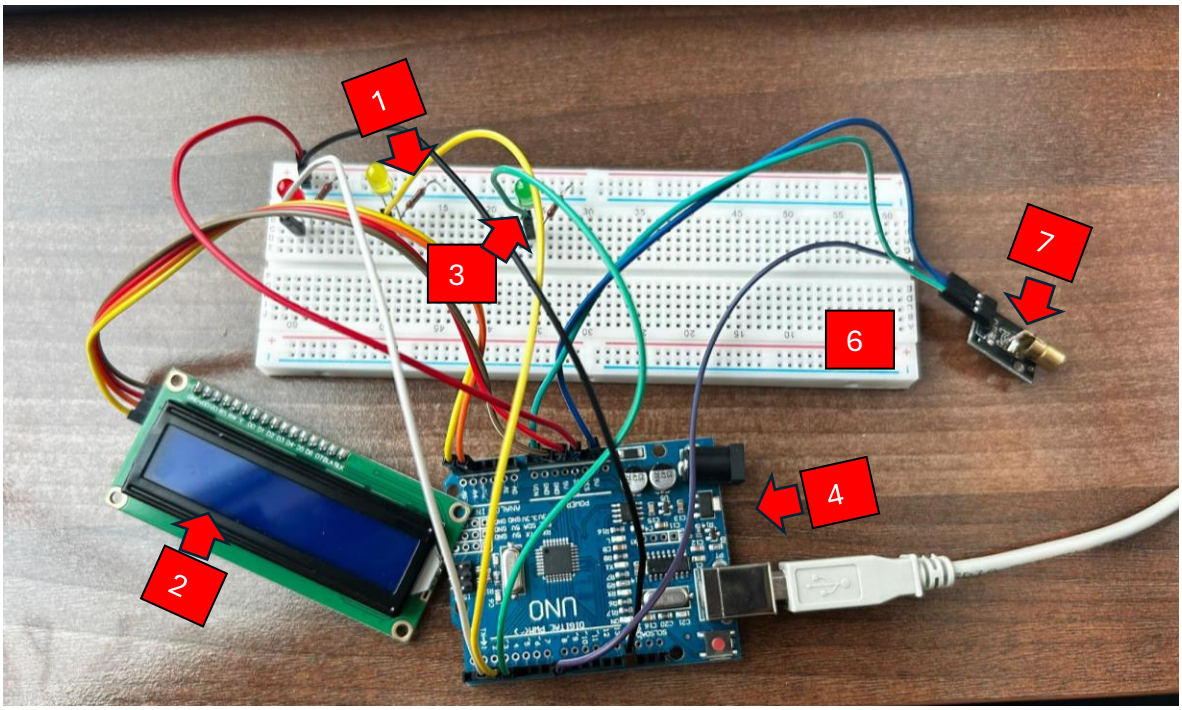


Figure 5.2: Full assembled system.

Key:

1. Resistors, 3 x 200 Ω for limiting current LEDs.
2. 16x2 LCD
3. 3 LEDs (Red, Yellow, Green)
4. Arduino UNO
5. Jumper wires
6. Breadboard with 2 power buses
7. KY-008 laser transmitter module.

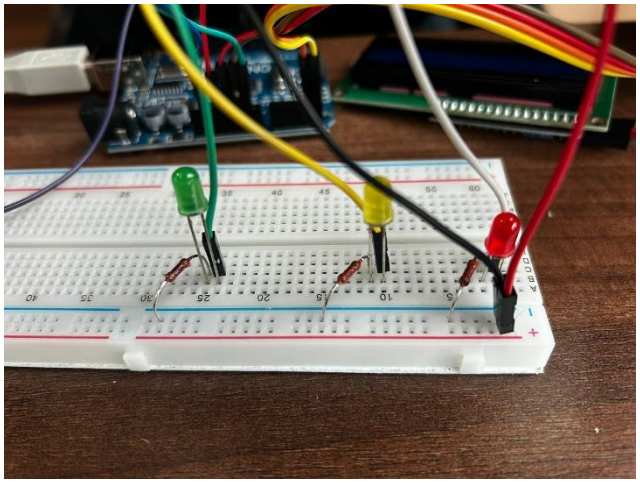

Figure 5.3: Close look from breadboard.

# 6 Software

For this project we used Arduino IDE (ver. 2.3.3) environment to program the Arduino UNO and implement the timing logic. The program reads input from the KY-008 laser module sensor and controls the LEDs and LCD display based on the timing process. The yellow LED indicates the system is ready, the green LED signals that timing is in progress, and the red LED shows the timer has stopped. The LCD displays messages to provide real-time feedback, such as "Counting...", and the elapsed time once the process is complete. This setup ensures accurate measurement of time intervals triggered by sensor activation, with clear visual and textual outputs provided by the LEDs and LCD.

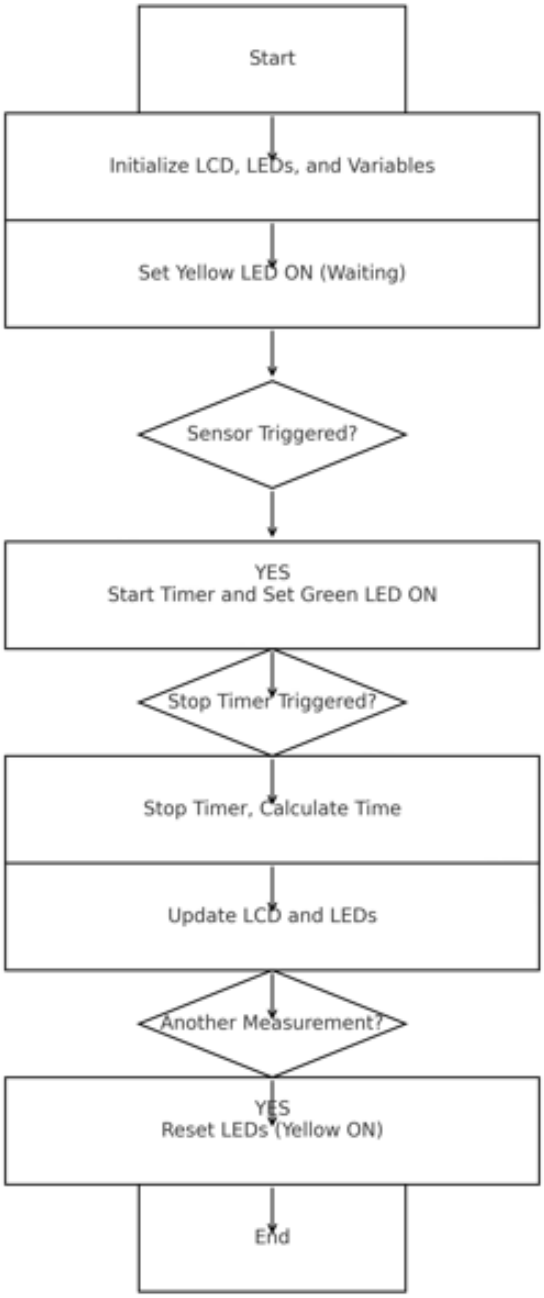


Figure 6.1: Block diagram of software.

The program begins by initializing the LCD, LEDs, and sensor pin, assigning them to the appropriate microcontroller pins. The LEDs are set to output mode, and the yellow LED is turned ON to indicate that the system is in a waiting state. The program continuously monitors the sensor and starts the timer when the sensor is triggered, updating the LEDs and LCD accordingly. When the timer is stopped by another sensor trigger, the elapsed time is calculated, displayed on the LCD, and the LEDs indicate the system's readiness for another measurement.

How the code looks:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>


LiquidCrystal_I2C lcd(0x27, 16, 2);


const int greenLED = 4;
const int yellowLED = 3;
const int redLED = 2;
const int sensorPin = 8;


bool timerRunning = false;
unsigned long startTime;
unsigned long endTime;

void setup() {

  lcd.init();
  lcd.backlight();

  Serial.begin(9600);

  lcd.setCursor(0, 0);
  lcd.print("Project Chrono");
  lcd.setCursor(0, 1);
  lcd.print("Initializing...");
  delay(3000);          // Hold the message for 3 seconds
  lcd.clear();


  pinMode(greenLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(redLED, OUTPUT);


  digitalWrite(yellowLED, HIGH);
  digitalWrite(redLED, LOW);
  digitalWrite(greenLED, LOW);
```

This part of the code initializes the hardware components and sets up the system for operation. It begins by including the necessary libraries for I2C communication and LCD control. The pins for the LEDs and the sensor are assigned to constants, and variables for timing are declared. In the “setup ()” function, the LCD is initialized, and backlight is turned ON, followed by starting serial communication for debugging. The LCD displays a startup message, "Project Chrono" and "Initializing...", which is held for 3 seconds before clearing the display. The LED pins are then configured as outputs, with the yellow LED turned ON to indicate the system is ready, while the red and green LEDs are turned OFF. This setup ensures the system is in a waiting state before any sensor activity occurs.

```
  // Initialize sensor pin
  pinMode(sensorPin, INPUT_PULLUP);

  // Set initial LCD message
  lcd.setCursor(0, 0);
  lcd.print("Ready...");
}

void loop() {
  // Read the sensor state
  int sensorState = digitalRead(sensorPin);

  // Debugging: Print sensor state to Serial Monitor
  Serial.print("Sensor State: ");
  Serial.println(sensorState);

  if (sensorState == LOW && !timerRunning) {
    // Start the timer
    timerRunning = true;
    startTime = millis();

    // Update LEDs
    digitalWrite(yellowLED, LOW); // Turn off yellow LED during timing
    digitalWrite(redLED, LOW);
    digitalWrite(greenLED, HIGH);

    // Update LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Counting...");
  } else if (sensorState == HIGH && timerRunning) {
    // Stop the timer
    timerRunning = false;
    endTime = millis();
    unsigned long elapsedTime = endTime - startTime;

    // Update LEDs
    digitalWrite(greenLED, LOW);
    digitalWrite(redLED, HIGH);
    digitalWrite(yellowLED, HIGH); // Turn on yellow LED to indicate ready state

    // Update LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Time: ");
    lcd.print(elapsedTime / 1000.0, 3); // Convert milliseconds to seconds
    lcd.setCursor(0, 1);
    lcd.print("Stopped.");

    // Debugging: Print elapsed time to Serial Monitor
    Serial.print("Elapsed Time: ");
    Serial.println(elapsedTime / 1000.0);
  }

  // Add a small delay to debounce the sensor
  delay(50);
}
```

This part of the code configures the sensor pin as an input with an internal pull-up resistor and sets the initial LCD message to "Ready...".

In the “loop ()” function, the sensor state is continuously read and printed to the Serial Monitor for debugging purposes. If the sensor state is detected as LOW and the timer is not already running, the timer starts, the yellow LED is turned OFF, the green LED is turned ON, and the LCD displays "Counting...". When the sensor state changes to HIGH and the timer is running, the timer stops, the elapsed time is calculated and displayed on the LCD, the green LED is turned OFF, the red LED is turned ON, and the yellow LED is turned ON again to indicate the system is ready for the next cycle. A small delay is added at the end to debounce the sensor input.

# 7 Start-up/Calibration

Our first attempt at this project was testing the LCD and later assembling the LEDs with the resistors we had (which was 10k ohm) and as expected it was not enough to light the LEDs as we wanted but we made sure that they were working properly.

Here is the testing code just to see if LCD working or not:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Hello, World!");
}
void loop() {}
```

The code demonstrated that our LCD was properly displaying everything we need.

**Sensor calibration and testing was later due to delay of delivery. That’s why sensor part was not added to this (start-up) section.

# 8 Test Measurements

In this project we didn't need too many calculations or measurements. Only calculation we ended up doing was resistor value for LEDs. The formula used was Ohm's Law:

$$R = \frac{V_{Source} - V_{LED}}{I_{LED}}$$

The output from the Arduino to breadboard was 5V and according to the researchers we had here is the simple chart for the LEDs values:

| LED | $V_{LED}$ | $I_{LED}$ |
|---|---|---|
| Red | ~1.8V to 2.2V | 0.02A |
| Green | ~2.0V to 3.0V | 0.02A |
| Yellow | ~2.0V to 3.0V | 0.02A |

The calculated value was 150 Ω. However, the reason we chose 200 Ω was because, we realized choosing slightly higher value reduces current and increases the LEDs lifespan while still providing enough brightness.

# 9 User Manual

Welcome to Chrono-BETA! The beta version of a chronometer so advanced, it's not quite ready for the world—yet. While our vision for Chrono-BETA involves precision laser-triggered timing, we're not quite there yet. Currently, Chrono-BETA is a charming two-person operation that still gets the job done, albeit with a touch of teamwork.

This user manual will guide you through the ins and outs of this quirky yet functional product. Let's dive in!

## What is Chrono-BETA?

Chrono-BETA is a versatile and entertaining chronometer designed to measure elapsed time with LED feedback and an LCD display. While its beta nature requires two people for operation, it still fulfills its primary purpose of timing events accurately—with a hint of collaboration.

Our future goal? Replace teamwork with technology! The planned upgrade involves a laser-triggered sensor to automate the start/stop functionality. But for now, Chrono-BETA embraces its human-powered charm.

## What's in the Box?

- 1x Arduino-based Chrono-BETA unit
- 1x LCD display (to flaunt your timing prowess)
- 3x Status LEDs:
    - Green: Ready
    - Yellow: Timing in progress
    - Red: Stopped
- 1x Photodetector sensor (aka, "The Helper-Dependent Sensor")
- 1x Set of jumper wires
- 1x Breadboard (for that retro prototyping feel)
- Instructions (that's this manual!)

## How Does It Work?

Chrono-BETA is simple to operate:

1. **Preparation Phase:**
    - Plug in Chrono-BETA to a 5V power source (or USB connection).
    - The LCD will greet you with "Project Chrono" and "Initializing..." before settling into "Ready..." mode.
    - Green LED lights up, signaling it's ready for action.
2. **Timing Phase:**
    - User 1 will manually block/unblock the photodetector to simulate the start/stop trigger.
    - The first trigger starts the timer, and the LCD switches to "Counting..." with the Yellow LED lighting up.
    - The second trigger stops the timer, displaying the elapsed time on the LCD.
3. **Results Phase:**
    - The Red LED lights up, indicating the timing has stopped.
    - The elapsed time appears on the LCD in seconds, formatted to three decimal places for precision (or as precise as teamwork allows).

## Troubleshooting

1. “The LCD isn’t displaying anything!”

Ensure all wires are correctly connected, especially SDA and SCL for the I2C interface.
Adjust the potentiometer on the back of the LCD to improve contrast.

2. “The LEDs aren’t lighting up.”

Check the resistors and connections for the LEDs.
Verify the Arduino code is correctly uploaded.

3. “The sensor isn’t working.”

Remember, this is Chrono-BETA! The sensor needs a little help to detect changes in light intensity. Try adjusting the light source or manually triggering it.

4. “The time isn’t accurate.”

Blame Person A for slow reflexes or upgrade to our future laser-triggered version! :D

Thank you for being part of this beta journey. The future of timing starts here!

# 10 Summary

The main idea of the project was to create a chronometer based on optoelectronic components. The elements we implemented included LEDs, a photodetector, and an LCD, considering their characteristics and behaviors. Although the original concept was to use a laser-triggered system for precise one-person operation, the current version relies on a manually assisted sensor. Despite this limitation, the system functions as intended, providing accurate timing results.

Developing the project came with its share of challenges, such as ensuring reliable sensor input, fine-tuning LED indications, and troubleshooting LCD functionality. Additionally, the manual photodetector operation meant that two people were required for use, which deviated from our initial vision. Nevertheless, we believe that with more time and resources, the inclusion of a laser module and enhanced automation could bring the project closer to its ideal state.

This prototype demonstrates the feasibility of using optoelectronics in timing systems and serves as a strong foundation for future improvements. The team's collaboration and shared vision allowed the project to be completed efficiently and on time, proving the value of iteration and problem-solving throughout the process.

# 11 Bibliography

[1] https://projecthub.arduino.cc/nimishac/make-your-own-stopwatch-with-arduino-bbcd1d
[2] https://www.dummies.com/article/technology/electronics/general-electronics/breadboard-led-circuit-223687/
[3] https://forum.arduino.cc/t/laser-beam-sensor/584642
[4] https://www.youtube.com/watch?v=af7_O6phL6E&t=581s